permitted to access during said learning period more leniently than attempts of the program to access files to which the user did not permit access during said learning period.

25. (Once Amended) A method for detecting abnormal behavior of a first application executed on a computer system, and preventing the damage thereupon, comprising:

- monitoring accesses of said application to elements of data storage arranged sectorwise in a storage device over a period of time and storing information about said accesses in an enforcement file, thereby learning the normal behavior of said application; and

- when said period is over, detecting attempts of said application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.

REMARKS

In the aforementioned Office Action, claim 19 was rejected under 35 U.S. C. § 102(b) as being unpatentable over Jablon et al. (U.S. Patent No. 5,421,006, "Jablon") and claims 21-34 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Jablon. Such rejections are respectfully traversed.

The present application:

In the present application the term "learning period"

refers to a period where there is a reasonable certainty that a software application is not infected by a virus. For example, assuming that a program was legally purchased, at the first period following installation of the program on a user's machine there is a high certainty that the application is not infected by a virus. During this period the accesses of the application to data storage locations (files, folders, etc.) are registered (referred in the present application as an "enforcement file"), and they will be considered as the normal behavior of the program. After this "learning period" ends, any future access to files by the application will be tested against the registration enforcement file, and accesses that do not correspond to the normal behavior observed during registration will be considered and treated as suspect.

The following description in the present application refers to the "learning" issue which is a central theme of the claimed invention: "... the system allows the attempt and learns the details so that in future an access to that area of the disk will always be allowed. Thus a specific enforcement file is gradually built up over the duration of the learn mode." (The present application, page 9, lines 15 to 17).

For example, during two weeks from the installation time (the learning period) of an accounting program, it will be assumed that all of its accesses were to the C:\MyAccounting folder. According to the present invention, after the learning period is over no accesses to any other folder will be allowed.

Additionally, according to a further aspect of the present invention, in order to prevent program A, whose access is

restricted by an "enforcement file", to invoke program B ("a daughter application" according to the present application terminology), which has no access limitations and therefore can reach any file/folder in the user's machine and cause a damage, the restrictions of program A are "heritated" to or imposed on program B. This subject is claimed, for example, in claim 30. For example, if a Web browser, which has an enforcement file, invokes a word processor, then the restrictions of the Web browser are applied to the word processor.

Thus, the present invention deals with the integrity issue by characterizing the normal access behavior of an application during a learning period, i.e., when the program is assumed to be uncorrupted, and restricting its accesses afterwards to what has been characterized as normal behavior.

Jablon:

Jablon deals with the integrity issue by indicating whether a program was altered since it was installed on a user's machine, and preventing changed programs from being executed. Indicating that a program was not altered is carried out according to Jablon by "verification data" (e.g., a digital signature) of the program: "... the first program verifies the integrity of the second program using the verification data stored in the protectable non-volatile memory." (Jablon, col. 8, lines 48-50). In order to keep the verification data away from a malicious object (e.g., a virus), the digital signature is stored in a "protectable non-volatile memory". Thus, integrity

according to Jablon means making sure that a program was not altered.

The major point of Jablon's invention is the "hardware latch memory protection mechanism" (Jablon, col. 7, Lines 16-17). Its purpose is described by Jablon as follows: "... the net effect of closing the latch is to prevent all subsequent programs from modifying the data .... BIOS will run the boot record program ..., and the system will continue its usual initialization process, with the restriction that the integrity assessment data cannot be modified. Thus, if the system subsequently changes the boot record, intentionally or not, the changes will be detected by BIOS the next time the system is reset." (Jablon, col. 12, lines 51-60).

The hardware latch memory enables storage of trusted data, whether it is a digital signature of the program or the program itself, and to restrict the access of said data to certain circumstances. "Before the first program transfers control to the second program, it computes, using known techniques, a modification detection code for the second program, and compares it to a pre-computed stored code in protectable non-volatile memory. If the codes are not identical, then the second program is not run, and the user is warned of the program." (Jablon, col. 8, lines 57-64).

"One or more regions of non-volatile memory are provided, in which security-relevant data are stored. Access to a protectable memory region is controlled with a latch mechanism, such that the memory is always both readable and writable when

the computer is first started. But during system initialization trusted software closes the latch to protect the memory, and thus prevent all subsequently run programs from reading and/or writing the security-relevant data during normal operation. Once closed, the latch can not be opened by software control. The latch is only re-opened when the system is restarted, which can occur by either momentarily turning off the power switch, or by pushing a reset switch. When the system is restarted, control of the CPU returns to a trusted startup program in read-only memory." (Jablon, col. 8, lines 18-33).

The differences between Jablon and the present invention:

1. Jablon doesn't use a learning process, i.e., a process on which the normal access behavior of the application is characterized.

2. In contrast to Jablon, in the present invention the target program is never examined in relation to verification data.

3. Jablon doesn't examine the behavior of the program, but the content of the program's file(s). Thus, while Jablon examines if the program was changed, the present invention examines its behavior. The present invention does not deal with indicating if a program was altered, but with indicating if the program's behavior is unexpected over what is considered as normal.

4. Jablon examines the program before its execution, while the present invention examines its consequences during its

execution. Thus, Jablon prevents the execution of a corrupted program, while the present invention prevents the damage thereof, not its execution.

5. And, contrary to Jablon, the present invention makes use only of software methods, not hardware methods.

The claimed invention:

Independent claims 19, 24 and 25 define a novel and unobvious system and method for detecting abnormal "behavior" of a software program and for preventing harm that might be caused by an abnormally behaving program. In this light, Applicant respectfully submits that the following underscores the salient distinctions between the Jablon system and the presently claimed invention:

a) While the term "integrity", as referred to by Jablon refers to an uncorrupted program. "Programs ... are verified before being utilized" (Jablon, the Abstract). In contradistinction, the term "integrity" as referred to by the present application is directed to the damage thereof. Thus, each application is directed to a different object.

b) While Jablon uses the enforcement device to prevent the execution of a corrupted program, the present invention uses the enforcement device to prevent the damage of a corrupted program during its execution.

c) In stark contrast to the present invention, Jablon doesn't involve or utilize any learning process. The only aspect of Jablon that, perhaps, can be considered as "learning" is calculating the verification data of the program, which is carried out at the installation time of the program. However, a calculation cannot be considered synonymous with machine learning. For example, machine learning is defined as "changes in [a] system that ... enable [it] to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.'' (Simon 1983, as cited at the following hyperlink: http://webster.cs.uga.edu/~miller/SemWeb/Presentation/ACTfiles/ml.ppt)

Machine learning is actually the obtaining of rules which characterize a population of samples of information of the same kind. However, Jablon doesn't deal with a population of samples, but with a single sample. He does not gather information from a plurality of uncorrupted programs in order to determine if another program is uncorrupted. Instead, he only gathers information of one program in order to determine if it has not been modified. Thus, Jablon doesn't use any machine learning.

Accordingly, Jablon does not disclose or suggest, either expressly or impliedly, the present invention is specifically set forth in Applicant's independent claims 19, 24 and 25. As such, those claims and the claims that may depend therefrom are believed to clearly and patentably distinguish the present invention from that taught by Jablon. Applicant thus kindly submits that the outstanding Section 102 rejection of claim 19

and the outstanding Section 103 rejection of claims 21-34, both in view of Jablon, are believed to be improper. Withdrawal of those rejections is therefore respectfully requested.

In view of the foregoing, the instant application is believed to be in condition for allowance and, therefore, early issuance thereof is earnestly solicited.

If the Examiner believes that a telephone interview would be beneficial to advance prosecution of the present application, the Examiner is invited to contact the undersigned at the telephone number listed below.

Respectfully submitted,

Date: Feb. 25, 2003

Christopher J. McDonald
Registration No. 41,533

Marked-up Version of Claims as Submitted by Amendment Dated Feb. 25 2003.

19. (Twice Amended) Apparatus for ensuring the integrity of an application executed on a computer having data storage arranged sectorwise, comprising:

apparatus for learning about the normal access behavior of said application to said data storage arranged sectorwise by monitoring accesses of said application to elements of said data storage during a limited period; and

an enforcement device, operative after said period is over, for identifying and preventing said application from accessing elements of data storage that do not correspond with the normal behavior of said application.

24. (Once Amended) Apparatus for ensuring the integrity of a computer application to be run in association with a computer having data storage arranged sectorwise in a storage device, comprising:

apparatus for assigning a general enforcement file to each new program;

apparatus for learning about the program by monitoring the [program of said data storage, by monitoring the program=s] program's attempts to make file accesses during a learning period;

an enforcement device operative, after said learning period is over, to treat attempts of the program to access files accessed during said learning period more leniently than attempts of the program to access files not accessed during said learning period[;], said enforcement device is based at least on instances of specific permission being given by the user to said

application to access locations of said data storage, wherein said enforcement device treats attempts of said application to access locations of said data storage to which the user has permitted to access during said learning period more leniently than attempts of the program to access files to which the user did not permit access during said learning period.

25. (Once Amended) A method for detecting abnormal behavior of a first application executed on a computer system, and preventing the damage thereupon, comprising:

- monitoring accesses of said application to elements of data storage arranged sectorwise in a storage device over a period of time and storing information about said accesses in an enforcement file, thereby learning the normal behavior of said application; and

- when said period is over, detecting attempts of said application to access elements of data storage that do not correspond to said normal behavior as determined by said enforcement file and inhibiting said accesses, thereby preventing the damage thereupon.